入门示例

此处提供一个播放器模块的基础使用示例，该示例包含以下内容

1. 根据直播状态，创建播放器观看直播或者单个回放视频
2. 调用播放、暂停等基础事件
3. 响应直播状态变化重新初始化播放器

更多内容请结合文档参考 观看页 SDK 及其配套的开源项目

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta
      name="viewport"
      content="width=device-width, initial-scale=1.0"
    />
    <title>Player</title>
    <style>
      html,
      body {
        padding: 0;
        margin: 0;
      }

      .demo-player-container {
        width: 640px;
        height: 360px;
        padding: 16px;
      }

      #player {
        width: 100%;
        height: 100%;
        position: relative;
      }

      .demo-player-control {
        display: flex;
        margin-top: 24px;
        gap: 12px;
      }
    </style>
  </head>
  <body>
    <div class="demo-player-container">
      <div id="player"></div>
      <div class="demo-player-control">
        <button class="demo-player-control__play-btn">播放</button>
        <button class="demo-player-control__pause-btn">暂停</button>
      </div>
    </div>

    <!-- 此处使用在线JS链接演示，开发者可使用npm包形式安装 -->
    <script src="https://websdk.videocc.net/live-watch-sdk/latest/lib/polyv-live-watch-sdk.umd.js"></script>
    <script>
      const {
        createWatchCore,

        PolyvWatchCoreEvents,
        ChannelEvents,
        PlayerEvents,

        LiveStatus,
```

```
} = window.PolyvLiveWatchSDK;

/** 全局变量 */
var globalStore = {
  watchCore: null, // 观看页核心实例
  playerInited: false, // 播放器是否初始化完成
  playStatus: false, // 播放状态
  liveStatus: LiveStatus.End, // 直播流状态
};

/**
 * 绑定 UI 事件
 */
function bindUIEvent(watchCore) {
  const $playBtn = document.querySelector('.demo-player-control__play-btn');
  const $pauseBtn = document.querySelector('.demo-player-control__pause-btn');

  $playBtn.addEventListener('click', () => {
    if (!globalStore.playerInited) return;

    watchCore.player.play();
  });

  $pauseBtn.addEventListener('click', () => {
    if (!globalStore.playerInited) return;

    watchCore.player.pause();
  });
}

/**
 * 绑定 watch-sdk 事件
 */
function bindWatchSdkEvent(watchCore) {
  // 监听核心 - setup 钩子
  watchCore.eventEmitter.on(PolyvWatchCoreEvents.WatchCoreSetuped, () => {
    console.info('观看页核心实例安装完成');
  });

  // 监听直播流变化
  watchCore.channel.eventEmitter.on(ChannelEvents.LiveStatusChange, ({ liveStatus }) => {
    if (globalStore.liveStatus === liveStatus) return;

    globalStore.liveStatus = liveStatus;
    console.warn('流状态变化', liveStatus);

    // 直播流变化时重新创建播放器
    if (globalStore.playerInited) {
      globalStore.playerInited = false;
      createPlayer(watchCore);
    }
  });

  // 监听播放器-初始化事件
  watchCore.player.eventEmitter.on(PlayerEvents.PlayerInited, () => {
    const playerInfo = watchCore.player.getPlayerInfo();
```

```
      globalStore.playerInited = playerInfo.playerInited;

      console.warn('播放器初始化完成');
      console.log('当前是否使用无延迟播放器：', playerInfo.isLowLatency);
    });

    // 监听播放器-播放信息变化
    watchCore.player.eventEmitter.on(PlayerEvents.PlayerInfoChange, ({ playerInfo }) => {
      globalStore.playStatus = playerInfo.playStatus;
    });

    // 监听播放器-播放时间改变
    watchCore.player.eventEmitter.on(
      PlayerEvents.TimeUpdate,
      ({ currentTime, durationTime }) => {
        console.warn('播放器时间变化：', { currentTime, durationTime });
      },
    );

    // 监听播放器-播放事件
    watchCore.player.eventEmitter.on(PlayerEvents.PlayerPlaying, () => {
      console.warn('播放回调，播放视频中');
    });

    // 监听播放器-暂停事件
    watchCore.player.eventEmitter.on(PlayerEvents.PlayerPause, () => {
      console.warn('暂停回调，已暂停视频播放');
    });
  }

  /**
   * 绑定事件
   */
  function bindEvent(watchCore) {
    bindUIEvent(watchCore);
    bindWatchSdkEvent(watchCore);
  }

  /**
   * 安装核心实例和授权
   * 注意：本示列使用无条件观看，故首次登录的用户信息会缓存有效期，如希望用户信息不缓存，请使用授权验证或者使用频道观看限制（如独立授权），二选一即可，如果使用授权验证，下面 频道观看限制 的判断代码就可以去掉，改成授权验证流程，建议带上await，然后再次执行await watchCore.setup() ，重新安装watchCore
   */
  async function setupWatchCore(watchCore) {
    await watchCore.setup();

    // 授权验证：https://help.polyv.net/index.html#/live/js/new_sdk/live_watch_sdk/articles/verify-next/overview

    // 频道观看限制
    if (!watchCore.auth.isAuthorized()) {
      // 观众点击观看页入口或执行观看条件授权，如：无条件授权
      const result = await watchCore.auth.verifyNoneAuth();
      if (!result.success) {
        console.error('授权失败了！！', result.failReason);
```

```
            return;
          }

          // 当观众执行观看条件授权成功，需要重新安装 watchCore
          await watchCore.setup();
        }
      }

      /**
       * 创建播放器
       */
      async function createPlayer(watchCore) {
        // 获取回放选项，此次仅以单个回放为例
        function __getPlayerPlaybackOptions() {
          const liveStatus = watchCore.channel.getLiveStatus();

          if (liveStatus !== LiveStatus.Playback) return;

          const playbackTarget = watchCore.playback.getSinglePlaybackTarget();
          watchCore.playback.setCurrentPlaybackTarget(playbackTarget);

          return playbackTarget.playbackOptions;
        }

        // 具体文档参数见：
https://help.polyv.net/index.html#/live/miniprogram/live_watch_miniprogram_sdk/articles/modules/player/player?
id=classmethoddoc_playermodule_setupplayer
        await watchCore.player.setupPlayer({
          container: '#player',
          lowLatency: false, // 是否无延迟播放
          showController: false, // 是否显示播放器控制栏
          playbackOptions: __getPlayerPlaybackOptions(), // 回放参数
        });
      }

      /**
       * 入口函数
       */
      async function main() {
        // 创建核心，文档见：
https://help.polyv.net/index.html#/live/miniprogram/live_watch_miniprogram_sdk/articles/core/sdk-core
        const watchCore = createWatchCore({
          channelId: 'xxx', // 请使用一个"公开观看"的频道
          userInfo: {
            // 用户信息
            userId: 'polyv-123456',
            nick: 'polyv-测试昵称',
          },
        });

        globalStore.watchCore = watchCore;

        // 绑定相关事件
        bindEvent(watchCore);
        // 安装核心
        await setupWatchCore(watchCore);
```

```
        // 创建播放器
        createPlayer(watchCore);
      }

      main();
    </script>
  </body>
</html>
```